A Portable C Compiler for the DEC-20

Jay Lepreau
Computer Science Department
University of Utah

April 24, 1981

Background

The primary research facility of the Computer Science Department is the Decsystem-2060, but many smaller machines are also heavily used, including several PDP-11's running Unix. One of the 11's is the primary machine used by the Department's well-known CAGD/graphics group. A VAX-11/750 is expected in June of this year, and promises to be of great importance to the various experimental research programs. Its large address space makes it particularly attractive to several groups who have exhausted their single-section address space on the 20, such as the LISP group and the Applicative Multiprocessing (AMPS) group.

Software support of all these machines has been a problem for the department, as no single higher-level language is available on all the systems. In the past this has sometimes required writing essentially the same program twice: once in C, and once in another language for the DEC-20. With the impending arrival of the VAX, such common software can be expected to become more prevalent, as users will expect equally rich environments on both the VAX and the 20.

This need for a common, compatible systems programming language prompted us to begin porting a C compiler to the DEC-20. There is already a C compiler on the 20, written by Alan Snyder of MIT [3], but it has a number of major deficiencies, the primary one being that it is not compatible with the current C language standard. It is also very slow and complicated, and thus a poor choice for modification.

The Portable C Compiler

We chose Steve Johnson's Portable C Compiler (pcc) [1] [2], which has served as the basis for C compilers on over a dozen machines. These machines cover a wide spectrum of architectures, ranging from 16 bit micros such as the Z8000, through highly irregular machines like the Harris/6 minicomputer, to the more regular, byte-addressed PDP-11 and VAX. The compiler is very robust and is capable of generating good enough code to serve as a production compiler, but the penalty for this code quality is that the first pass is not entirely machine independent.

Current Status

The compiler on the DEC-20 has been capable of compiling itself for about two months, and successfully compiles the approximately 30,000 lines of arbitrary C programs in our test suite. It is already being used successfully by at least three different research groups in the Computer Science Department, in part because of its complete compatibility with the C compilers on our PDP-11's. However, it still needs a significant amount of work to be a production compiler, namely in the following areas:

- Characters are currently stored one per word. This is only a temporary measure, for most of the code laying the groundwork for packed characters has been written. This was the major difficulty in porting the compiler, as no real provision had been made for a word-addressed machine.

- The compiler needs tuning to improve the quality of the generated code. The addition of a peephole optimizer would be the most effective way to significantly improve the code, and we hope to use either an existing PDP-10 optimizer or a purportedly portable one developed at the University of Arizona.

- Bit fields are not yet supported by the code generation routines.

- Double precision floating point operations and variables are not supported.

In addition to the improvements needed to the compiler itself, work is required on the associated subroutine library. Since the C language is

similar to Algol in that no input/output facilities are defined by the language, but are instead provided by subroutines, the provision of a substantial library is essential in order to furnish a usable environment. A standard set of these library functions has long been defined as a C standard, and work is in progress to implement this set on the DEC-20. We are putting great emphasis on compatibility with the existing Unix library in order to encourage programs which are as portable as possible. Results to date have been encouraging, and due to the flexible and powerful nature of the TOPS-20 operating system we see no obstacles to providing this compatible environment.

Future Work - Extended Addressing

If all of the above go well, we will have achieved our goal of a compatible language and library for our environment. However, one problem remains: the DEC-20's limited address space relative to the VAX, our other major machine. A general-purpose language which supported extended-addressing would greatly extend the research capabilities of the 20 as well as provide true compatibility between the VAX and the 20. The portable C compiler, with its machine-dependent modules and functions cleanly defined, seems the ideal candidate. Having worked with the compiler for some time, we have now gained the intimate familiarity with its structure which is essential in making this a realistic goal.

Some of the most difficult problems with extended addressing appear to lie with lack of support by the monitor. In addition, there are efficiency problems arising from the 20's inadequate caching during virtual address translation. If these problems make it too difficult to immediately achieve our full goal of complete extended-addressing, it would still be very useful to have multiple data sections, which would avoid some of the more obvious problems.

These issues will be identical for anyone attempting to implement extended-addressing, and must be addressed by all. Thus it will be to our mutual advantage that Dr. Martin Griss, in this department, will be simultaneously implementing an extended-addressing LISP. Both of these projects will benefit from the common expertise gained, and the duplicate effort certain to be saved.

REFERENCES

[1] Johnson, S. C.
A Tour Through the Portable C Compiler.
Technical Report, Bell Laboratories, Murray Hill, N.J., 1977.

[2] Johnson, S. C.
A Portable Compiler: Theory and Practice.
In Conference Record of the Fifth Annual ACM Symposium On Principles of Programming Languages, pages 97-104. ACM, January, 1978.

[3] Snyder, A.
A Portable Compiler for the Language C.
Project MAC TR-149, MIT, Cambridge, MASS, 1975.